DAHLGREN DIVISION
NAVAL SURFACE WARFARE CENTER
Dahlgren, Virginia 22448-5100

NSWCDD/TR-01/134

INITIAL REQUIREMENTS FOR A SOFTWARE TOOL TO SUPPORT THE USE OF OPERATIONAL SEQUENCE DIAGRAMS (OSDs)

BY DR. DANIEL WALLACE (NSWCDD)

JOHN WINTERS
(BASIC COMMERCE AND INDUSTRIES, INC.)

WEAPONS SYSTEMS DEPARTMENT

NOVEMBER 2001

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	November 2001	Final

4. TITLE AND SUBTITLE

INITIAL REQUIREMENTS FOR A SOFTWARE TOOL TO SUPPORT THE USE OF OPERATIONAL SEQUENCE DIAGRAMS (OSDs)

5. FUNDING NUMBERS

6. AUTHOR(s)
Dr. Daniel Wallace (NSWCDD)
John Winters (Basic Commerce and Industries, Inc.)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Commander
Naval Surface Warfare Center
Dahlgren Division (Code G50)
17320 Dahlgren Road
Dahlgren, VA 22448-5100

8. PERFORMING ORGANIZATION REPORT NUMBER

NSWCDD/TR-01/134

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
In order to produce systems to meet the needs of the end users, frequent and detailed inputs and feedback from the user community and domain experts should be integrated into the development process. Current methods for soliciting user input are largely anecdotal or reactive, but a structured tool to document and organize such information can permit the systematic integration of the real end-user requirements. Based on experiences in using a modified version of operational sequence diagrams (OSDs) on several projects at the Dahlgren Laboratory of the Naval Surface Warfare Center, Dahlgren Division (NSWCDD), proposed requirements for a software tool to support the use of OSDs were produced as part of the ONR/SC-21 Science and Technology Manning Affordability Initiative.

This report documents the initial requirements for a software tool to support this process. An introductory description of the revised OSD symbology is also provided.

14. SUBJECT TERMS
Operational Sequence Diagrams, Task Analysis, Function Analysis, User Requirements, Human-Centered Design, Human Factors Engineering, Subject Matter Experts

15. NUMBER OF PAGES
36

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500
Standard Form 298 (Rev 2-89)
Prescribed by ANSI std. Z39-18
298-102

FOREWORD

Participation of the user community in the design process is critical to developing systems that fit the needs of the users. The earlier such participation occurs, the greater the potential for positive impact. Requirements must be collected from the user community in a language they understand, and proposed designs must be presented for feedback.

A modification of the traditional operational sequence diagram (OSD) format has been employed at the Dahlgren Laboratory of the Naval Surface Warfare Center, Dahlgren Division (NSWCDD), on several projects, and it has been found to be a very effective means of soliciting input from users and domain experts. This report outlines the basic principles of this revised OSD methodology. The revisions and requirements explained in this report were performed under the Human-Centered Design Environment (HCDE) thrust of the ONR/SC-21 Science and Technology Manning Affordability Initiative.

As effective as OSDs have been, their ability to contribute to the design process can be improved even further with implementation in a software tool. Such a tool will enable the solicitation and integration of user input and feedback in a manner that is faster, more efficient, and more frequent.

Approved by:

DANNY BRUNSON, Head
Weapons Systems Department

CONTENTS

ILLUSTRATIONS

INTRODUCTION

Three decades ago, Kurke proposed a symbology for defining system operational sequence.[1] As a systems engineering tool, the application of this symbology is perhaps the most powerful technique available to the human factors practitioner.[2] Since its creation, this technique and its variants have been used in many system development efforts as tools for both analysis and communication. One primary variant is specified in the U.S. Defense Department's handbook, MIL-HDBK-46855A. Kurke's original version and this variant are shown in Figures 1 and 2. The primary differences between these earlier operational sequence diagrams (OSDs) and those currently in use are the orientation and column format. Earlier versions of OSDs were primarily used for detailed analysis of the human-machine interface, with tasks performed by different humans or other system components shown in different columns. The revised version does not use such a format, permitting it to be extended for use in design stages during which allocations to humans or machines are not yet able to be defined.

Because of the distinctive system requirements of every project encountered, most human factors practitioners have tailored OSDs to meet their unique needs. Chapanis and others have demonstrated the broad range that pictorial representations of system definition can take and still be legitimately called OSDs.[3,4] Other broadly comparable methodologies include Integrated Definition (IDEF) Models, Decision/Action Diagrams, and Functional Flow Diagrams; however, these do not always capture the full range of information captured in OSDs.

REVISED OSDS

The current OSD format was created during 1993-94 when the U.S. Navy directed a rearchitecture of a command and control system for a particular ship self-defense system. A major concern for that effort centered on the human factors, including how to effectively define the human operator's role in the system so that it would be understandable to such varied populations as the system engineers, the Fleet users (subject matter experts (SMEs)), and the human factors team.

OSDs were the most viable option for this effort, but the traditional format proved problematic in the following areas:

- Readability/understandability for SMEs
- Ease of creation, review, and modification
- Ability to derive workload parameters
- Linking information to specific tasks/decisions

Initial interviews with Fleet SMEs demonstrated that, because of prior experience with flowcharting procedures, a more "flowchart-like" representation of the system processes provided a more easily understood medium for the Fleet reviewers. Similarly, such a format proved to be a more effective communication tool to convey anticipated system operations to the developer community as well.

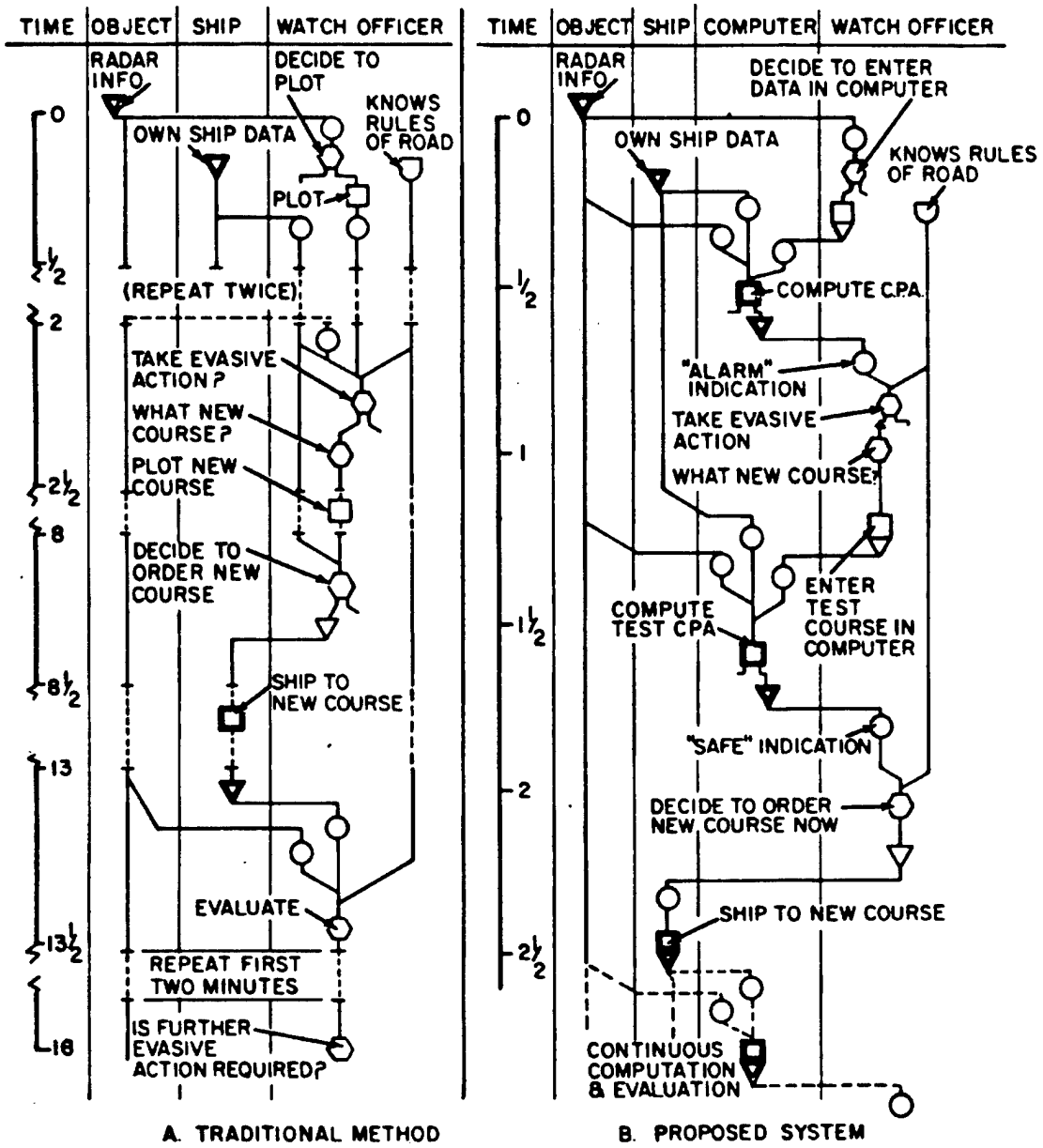

Figure 1. OSD Example from Van Cott and Kinkade[5] (Originally in Kurke[1])

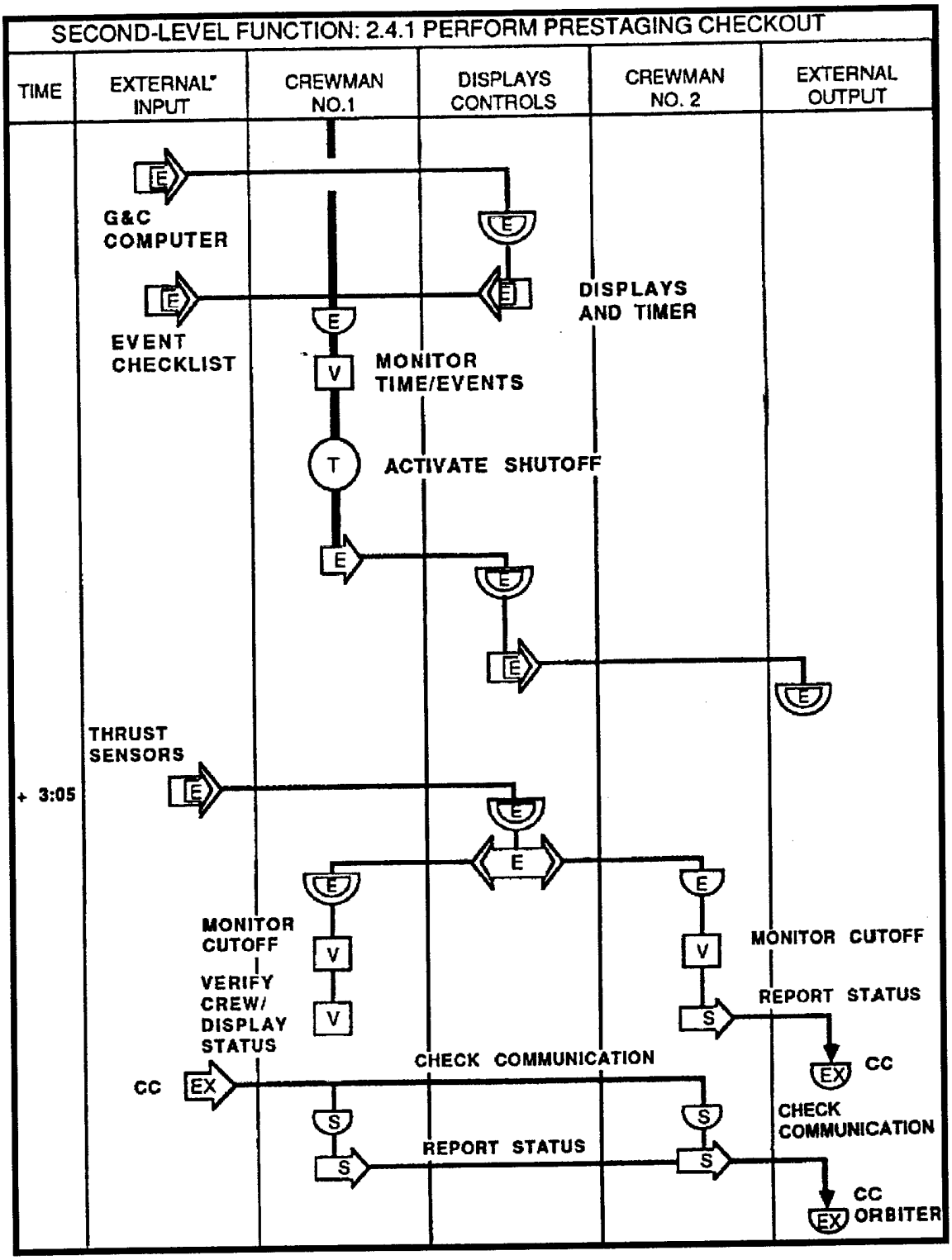

Figure 2. OSD Example from MIL-HDBK-46855A, p. 134[2]

This revised OSD methodology was subsequently used on a variety of programs with a variety of different goals. Some of those projects are listed below:

- **Rearchitectured NATO Sea Sparrow Missile System (RNSSMS):** The redesign of the NSSMS required a full understanding of the information requirements associated with all of the actions performed by the system and operators. The OSDs provided a means to solicit, capture, and review the requirements associated with the expanded capabilities of the RNSSMS. These also provided a means to manage and review the continued evolution of design concepts.
- **Joint Interface Control Officer (JICO):** Following the development of a task analysis for the JICO organization, OSDs were developed to more fully document the operational flow and to associate information requirements with each of the JICO functions. These OSDs were then used in the development of a Systems Requirements Document that identified core requirements for a JICO Support System.
- **Area Air Defense Commander (AADC):** As with the JICO project, OSDs were developed as a follow-on effort to the task analysis work. These OSDs helped identify the relationships between functions performed by the AADC and helped form the basis for the AADC Systems Requirements Document. Human Factors Engineers working on the production AADC system are now using these OSDs to assist in the development and validation of the system design.
- **Navy Manpower Analysis Center (NAVMAC):** The initial step toward designing a future analysis tool for NAVMAC was to analyze the current processes and then derive the process for future operations. Current Process OSDs were built using SMEs and previous IDEF diagrams. Meetings were held with SMEs and current analysts to determine and document the vision for future analyses. The OSDs demonstrated areas in the process with the largest return for software development and defined the interfaces to existing software.
- **Manning Affordability Initiative, Human-Centered Design Environment (HCDE):** OSDs were used to document task analyses of systems engineering and human engineering processes. These were used to identify information and tasks that were common to the two processes.
- **Supporting Arms Coordination Center (SACC) Reconfiguration:** A simplified version of the OSD components was used to describe how a Call For Fire is received, processed, and responded to by personnel in a SACC. These diagrams were used as references in creating a task network model in the Integrated Performance Modeling Environment (IPME) of the Call For Fire process.

Usage and evolution of the methodology on these programs demonstrated the effectiveness of the resulting OSDs to address not only the elicitation and review of system capabilities and requirements from the user community, but to fill that critical role of identifying human-driven system requirements for the developer community. These factors span the gamut from function allocation and workload estimation to actual interface design. The resulting system requirements and system capabilities definition are now convertible into the "spec language" required to define system specifications for acquisition. This iteration of the OSD methodology is described in Wallace, Bohan, and Perry (see Reference 6), and the arrangement of elements into a sample Task Unit is shown in Figure 3.

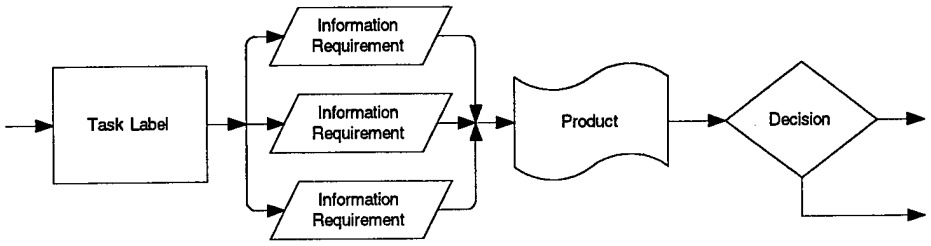

Figure 3. Sample Task Unit, Interim Version of OSDs

Despite the utility and effectiveness of the OSDs, it was found their format had a number of shortcomings. SMEs found that, although they were able to learn to read the new format faster than the more traditional OSD format, it was still somewhat difficult to learn. Where there is only a one- or two-day window for SME review, this overhead can be problematic. There were also some misunderstandings regarding flow within a task unit as opposed to flow between task units; many SMEs, by reading the line between the task box and the information requirements box, were confused as to why one was executing a task before getting all required information. Lastly, reviewers found it difficult to focus on specific tasks because the other information was distracting.

Due to these findings, alternative arrangements of the OSD elements were explored, with the intent to maintain the basic elements of the diagrams. The symbols for individual diagram components were not altered, but their arrangement into Task Units was refined. Different options were developed, and two versions were evaluated by subjective preferences and observations in the creation of new diagrams using each of the options. Reviewers included those with and without prior exposure to the earlier version of the OSD methodology. The details of this review process can be found in Wallace, Winters, and Lackie.[7]

The symbols for individual elements of the revised version of the OSDs are shown in Figure 4. Used together, these elements allow the construction of Task Units, as shown in Figure 5. Two variants based on the arrangement of task-related decisions are shown in Figure 6. The revision reduces the problems associated with distinguishing between flow within a Task Unit and flow between Task Units by using a different style of line ending for horizontal lines within a single Task Unit. Flow is not meant to be implied within a task unit; flow is implied by arrows external to a task unit. Definitions of each of the major elements are as follows:

- **Task Label:** The fundamental element of the OSD defining the principal task, action, or operation performed. It is the anchor for the Task Unit.
- **Information Requirement:** Also used to define any system capabilities required to execute the Task Unit.
- **Decision:** Used to route operational sequence and task flow.
- **Product:** Denotes any product produced by the Task Unit. This could be a physical item, a data product, or any other definable output.
- **Trigger:** Defines any events external to the system that can initiate a Task Unit.
- **Router:** Defines flow between different diagrams within a single diagram set.

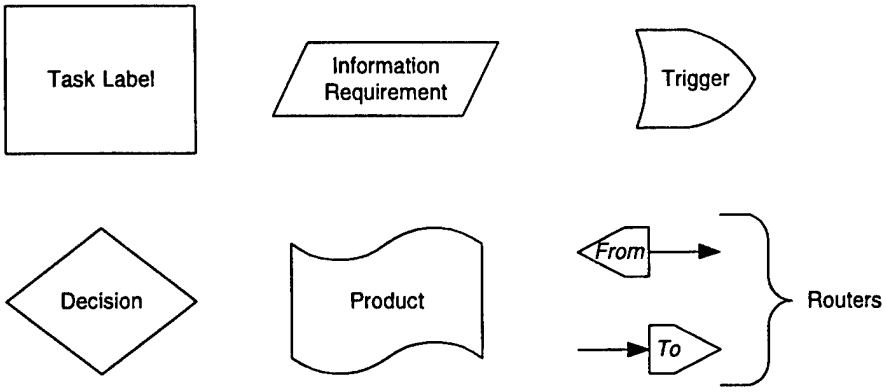

Figure 4. Major Components of an Operational Sequence Diagram

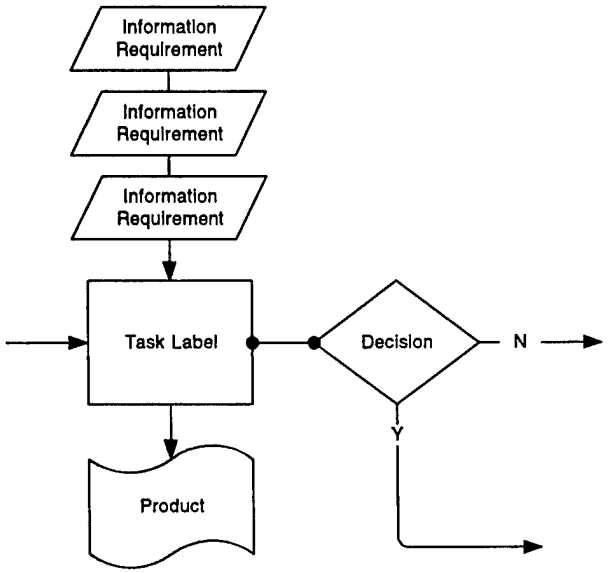

Figure 5. Sample Task Unit

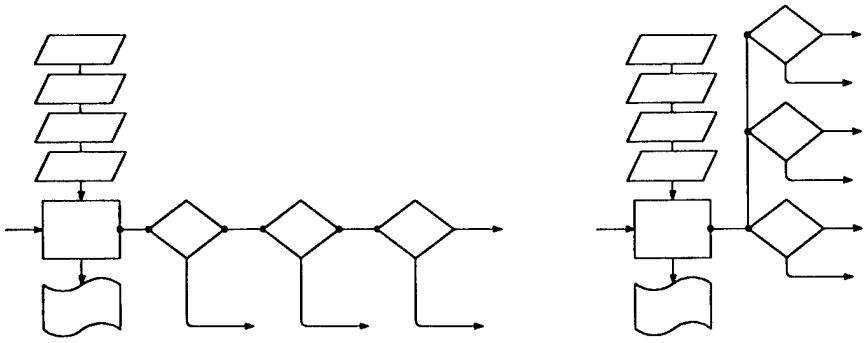

Figure 6. Task Units with Serial (Left) and Parallel (Right) Decisions

As shown in Figures 5 and 6, operational flow enters the Task Label from a triggering event or another task. Then, any information requirements are explicitly identified. If there are any products, those are so indicated, and likewise for any decisions that are associated with the task. After the total system is defined using the OSDs, decisions can be made regarding function allocation. The convention of a double line border for each element used in more traditional OSDs can be employed here as well (though not shown).

In this section and the sections that follow, the term *diagram* refers to a contiguous sequence of Task Units and other diagram elements, separated from other diagrams by User-defined transition points. The term *diagram set* refers to the collection of diagrams created for an entire system or project. The relationships between a diagram set, diagrams, Task Units, and other components are shown in Figure 7.

SOFTWARE TOOL TO SUPPORT OSD METHODOLOGY

When utilized in the projects listed in the previous section, OSDs were created using commercial diagramming tools. While these tools are relatively inexpensive and flexible in their graphical capabilities, they are not interoperable with the engineering tools. The ability to transfer information such as design scenarios, task flow, and information requirements is lacking. SME input cannot be easily imported into an engineering tool to create a function or task flow, and existing task and function models cannot be easily provided to SMEs for review. Reviewers are forced to use reports and outputs designed for engineers and designers to provide feedback on designs. Better feedback could be provided by SMEs if they are provided with a task flow or scenario representation rather than a stack of requirements documents or engineering details.

An integrated OSD tool would allow SMEs to document operational scenarios and task flows that, if the level of detail is appropriate, can then be easily imported to engineering tools to form the basis for executable simulations. Additionally, existing simulations can be provided to SMEs for review and feedback in a format they can better understand in a tool that is easier to use. The flexibility of commercial packages can be a detriment in that standardization is practically impossible and in that specialized functionality is difficult to access. The added functionality of engineering tools is inappropriate for most reviewers since such tools are typically more expensive, require greater training, and do not lend themselves to the review process. An integrated OSD tool would allow rapid transition from basic task flows and scenarios to executable simulations and provide for structured feedback from reviewers who are experts in operational aspects but not in engineering or design.

The requirements presented in this document are partially based on the assumption that the existing OSD symbology and methodology will be used to represent information. Deviation is acceptable, but it should only be based on a demonstrated improvement in capability or usability or due to technical limitations. The design of the tool should be sufficiently flexible to allow for minor modifications or additions to provide for the needs of individual projects. Flexibility from project to project will be required. One example of such flexibility would be the ability to support other diagramming approaches. Some diagramming methods may be supportable with only minor modifications. If, for example, Products and Information Requirements were omitted, the diagrams would be functionally equivalent to Decision-Action Diagrams.

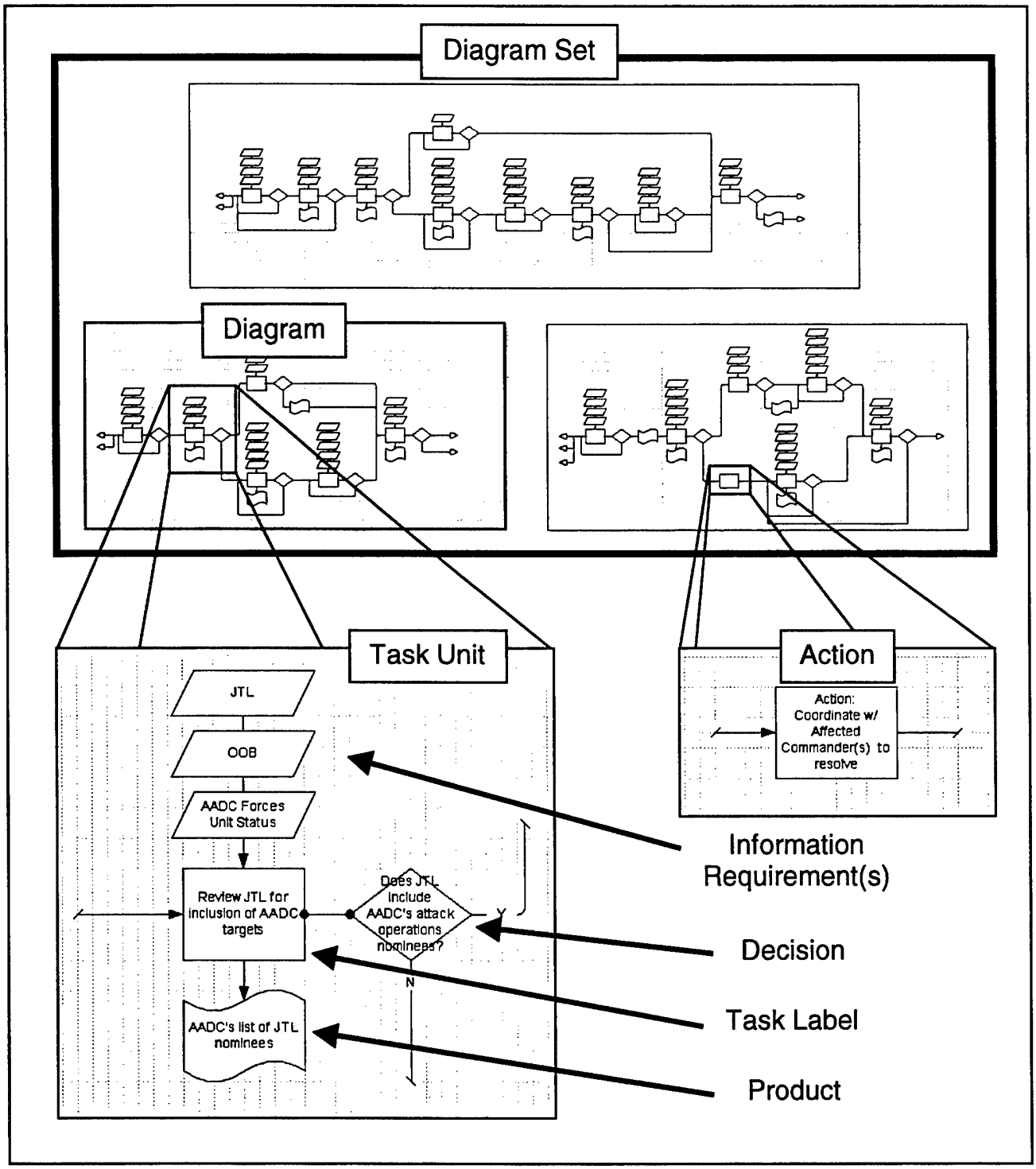

Figure 7. Hierarchy of Diagrams and Diagram Elements

INTENDED USERS

The primary users can be separated into two categories – Analysts, who have expertise in the definition and analysis of systems and their requirements, and SMEs, who have experience with and knowledge of the domain being modeled or analyzed. In addition to these two primary types of users, those who employ the results or output of the tool to create models or simulations comprise a third category. These simulations may include those for the overall system, or there may be an emphasis on task modeling. This category of user may also include the creators of models that are reviewed and commented upon through the use of the tool.

All users are assumed to have prior experience with basic personal computer operations, including file management in Windows (or Windows-like) operating systems. The majority of users are expected to be proficient in basic capabilities of common word processing software packages.

ANALYST

The Analyst is familiar with techniques and procedures for task analysis or scenario documentation. A goal of the Analyst is to elicit and document subtleties of system operation that are implicit in the minds of SMEs. A second goal is to uncover the true, underlying requirements for the system, as opposed to implementation-specific requirements that are more easily articulated by the end-user community. For this User, the tool is primarily a timesaving device that allows the rapid creation of diagrams and documentation of associated information. The features allowing integration with other design tools are also valuable, as they permit the Analyst's work to be easily transferred to other tools for additional use. The Analyst can be expected to be a frequent and recurring user of the tool, employing it in the design of multiple systems or documentation of a variety of scenarios. Of the three groups of users described here, the Analyst will be the most proficient user of the tool and is likely to have prior experience with other task modeling or diagramming tools and methodologies.

SUBJECT MATTER EXPERT

The Subject Matter Expert (SME) possesses detailed knowledge and understanding of the operational context for which the system is being designed, and that knowledge and understanding are put to use in either creating or reviewing descriptions of how a system might operate or a scenario unfold. The SME may use the tool personally or with an Analyst who supports data capture and knowledge elicitation functions. The SME is involved in the design process due either to knowledge of the goal or intent of the system or to experience as a User of the system or a similar system.

The SME is not expected to be proficient in the use of modeling or simulation tools. Therefore, the tool must be simple enough to allow an SME to quickly learn its basic functionality through trial-and-error use of the tool or a very brief demonstration.

MODELER

The Modeler may or may not have a comprehensive understanding of task analysis techniques and objectives. Additionally, the Modeler is not expected to have detailed knowledge about the domain for which the given system is being designed, although he or she may have significant knowledge and understanding of the design of the system itself.

The Modeler is usually an indirect user of the tool, making use of its outputs and creating inputs for the tool. The Modeler will use scenarios or task flows created by Analysts or SMEs as initial models within a modeling tool. The Modeler will be required to access and interpret notes and descriptions within the imported information to determine how to complete an executable model of the system being designed. The Modeler will also provide a current version of a model for import by the tool. The translated model will then be sent to reviewers for feedback and comments. The Modeler may be a direct user of the tool in order to review comments and information provided during the review process. If the modeling tool used is incapable of receiving all of the information associated with Task Labels and Task Units, then the Modeler may be required to continually review the diagrams in the OSD tool to access information required to build a more effective and accurate model.

USAGE SCENARIOS

The OSD tool must be sufficiently flexible to support a wide variety of uses, including creation of diagrams, review of diagrams, and sharing of information with other design tools. Three primary usage scenarios for the OSD tool have been defined in order to illustrate how the different users might employ the tool to accomplish their tasking.

KNOWLEDGE ELICITATION AND REQUIREMENTS DEFINITION

The OSD tool is employed to generate background information necessary to define an initial set of requirements for a system. The tool will be used to document potential operation of the system within its intended environment by a set of SMEs. Following a one-hour orientation briefing on the use of the tool and purposes of the project, the SMEs divide the operation scenario into its major elements. The major elements are entered into the tool directly, creating a wiring diagram and a blank diagram for each of the major elements.

The SMEs then assign each of the separate diagrams to an individual. Working individually, each SME defines the details of each major element, occasionally dividing a single diagram into multiple diagrams. At first, the SMEs use the knowledge elicitation capabilities of the tool to create the diagrams. They are asked by the tool to first identify the discrete tasks within the process. For each individual task, they are prompted to identify information or other inputs required for the task, relevant decisions, and potential products. They are able to browse the list of Products from the entire diagram set to identify specific inputs required for the tasks. This information is all entered in a question-and-answer format, and the tool generates the graphical version of the diagrams automatically. Once generated, these diagrams can then be directly modified. As the SMEs become more comfortable with

the process of generating diagrams, they begin to create Task Units and define their elements directly instead of using the knowledge elicitation capability.

During their weekly meetings, the diagrams created by each participant are automatically combined into a new set of diagrams. The overall wiring diagram is updated automatically. The group discussion time is used to identify relationships between diagrams that are the responsibility of different individuals. These types of relationships include sequential flow between diagrams and identification of common Products or Information Requirements. In case questions about another individual's diagrams arise between these meetings, the individual responsible for the definition of each diagram is listed in the electronic version of the diagram set.

Following the completion of the diagrams by the initial set of SMEs, the diagrams are reviewed as a group by an additional set of SMEs. Following recommended modifications, the tool is used to export the information in the diagrams in a format that can be easily transitioned to the initial requirements document for the system to be designed.

TASK DESIGN

An Analyst employs the OSD tool to document observations made during a task analysis of an existing system. The tool is also used to design tasks for additional functionality to be designed into the next version of the system. The Analyst creates the first version of the diagrams, and an SME provides input and feedback on an ongoing basis.

Once the initial version of the diagrams is complete, the Analyst distributes them to a set of SMEs for their review. Using commenting features of the tool, the group provides comments on a subset of the diagram set. The remaining diagrams are reviewed individually by one or more of the SMEs. Not only are the SMEs able to add comments to individual portions of the diagrams, but they are also able to suggest reordering of the task flow, add and delete Information Requirements and Products, and edit the diagrams in general. The Analyst receives reviewed copies of the diagrams from different SMEs, but the OSD tool compiles the comments in a single file for the Analyst. Using this feedback, the Analyst is able to determine what comments and recommendations have been provided and updates the task analysis accordingly. The reviewed and edited diagrams are then exported from the OSD tool for use in a task network modeling tool.

VALIDATION OF REQUIREMENTS AND DESIGNS

A requirements document has been used by a group of Modelers as the basis for an executable model of a proposed system. To assist in the validation of their model, they export their model in a format readable by the OSD tool. The tool is then used to prepare materials for review of the proposed system at a workshop with a dozen SMEs. With the assistance of one of the Modelers, an Analyst walks the SMEs through the operation of the proposed system. Comments and suggested alterations are recorded as they surface using the OSD tool.

The team of Modelers then has the option of either combining the modified and approved set of diagrams with their previous model or creating a report from the tool that describes all of the modifications and amplifying information added to the diagrams. Since

the elements in the operational sequence diagrams share common identifiers with the elements of their model, they are able to ensure that all recommendations are either followed or provide a rationale for deviating from the recommendations.

REQUIREMENTS

The following sections provide details about anticipated tool requirements in the areas of information representation and support for user tasks and goals. Within each subsection, background information on the use of the proposed tool is provided. The specific requirements that relate to that area of functionality are provided as a bulletized list. These requirements are based on use of operational sequence diagrams on a variety of projects and on a comparative evaluation of different modifications to the symbology, but they are preliminary requirements only. Further refinement and expansion will be necessary as details of tool implementation and user scenarios are extended.

INFORMATION REPRESENTATION REQUIREMENTS

The foremost requirement of a software tool to support the creation and usage of operational sequence diagrams is that the tool be able to adequately display appropriate elements of the diagrams. The requirements listed in this section deal with how information should be represented, both graphically and textually. Precise compliance with the symbology and conventions of the previously described revision to the OSD methodology is not required, but that methodology is assumed to satisfy all of the information requirements provided in this section.

Task Units

The fundamental building block of operational sequence diagrams is the Task Unit. The Task Unit is a combination of different elements that specify details of the task represented – the information that is required, any decisions made in the course of the task, and any products generated. Although the individual elements of the Task Unit must be apparent and understandable, the Task Unit itself must be a coherent unit. The requirements for each component of the Task Unit are outlined in the following subsections. (*See also the Diagram Creation section under Task Support Requirements for related information.*)

- The extent of the Task Unit as a whole (Task Label and all associated Information Requirements, Products, and Decisions) shall be graphically apparent, being easily discerned from other Task Units and diagram elements. It is anticipated that the graphical format outlined in this document satisfies this requirement.
- The User shall be able to add, delete, and edit individual Information Requirements, Products, and Decisions without having to make other changes to the Task Unit.

Task Labels. Each Task Label serves as an anchor for the Task Unit of which it is part. The Task Label defines the principal task, action, or operation performed. Although there may be multiple Information Requirements, Products, and Decisions in a Task Unit, there is only one Task Label per Task Unit.

- Task Labels shall be graphically associated with the Information Requirements, Products, and Decisions that are included in the Task Unit.
- The labeling of the Task Label shall be a shorthand description of the activity that the Task Unit represents.
- The User shall be able to record a longer and more detailed description of the task that is associated with the Task Label but is not displayed with the graphic representation of the Task Label.

Information Requirements. Information Requirements represent the pieces of information required to execute the task. The information is typically designated by its title or name, not a specific value. For tasks with material requirements, similar nomenclature can be used to represent inputs to the task.

- Information Requirements shall be graphically associated with the Task Unit for which they are employed.
- Information Requirements that were created internally to the current set of diagrams shall be graphically distinguishable from those created outside the current set of diagrams. (Information Requirements that are a Product of a Task Unit elsewhere in the diagram set are termed *internally created*. Information Requirements that existed prior to the activities depicted in the diagram set or that are created outside of the documented activities are termed *externally created*.)
- The labeling of Information Requirements created within the current set of diagrams should include the location (diagram number at minimum) at which the Information Requirement was created (i.e., was a Product of another Task Unit).
- The default list order of Information Requirements within a Task Unit shall be externally created items, listed alphabetically, followed by internally created items, listed according to the order in which they were created. The User shall be able to change the default order for an individual Task Unit if desired.
- Due to the possibility of task iteration, the User shall be able to specify Information Requirements that have not yet been created within the basic flow of the diagrams. (Between the first iterations and a subsequent iteration of a Task Unit, an additional Information Requirement may have been created elsewhere. This Information Requirement would not be available during the first iteration but would be available for later iterations. This type of Information Requirement must be specifiable by the user before the Task Unit that will create the Information Requirement has been documented.)
- When entering details about a Task Unit, the user should be able to (but not be required to) select Information Requirements from a list of existing Information Requirements and Products for other Task Units. The existing Information Requirements and Products shall be sortable by name, by diagram, and by date created.

Products. A Product is any tangible output produced by a Task Unit. Products include physical items, data products, and any other definable output. A Product may subsequently be used within the same diagram, within another diagram in the same diagram set, or externally to the diagram set altogether.

- Products shall be graphically associated with the Task Unit within which they were generated. A Task Unit may have no Products, a single Product, or multiple Products.
- Positioning and layout options shall provide the ability to clearly show whether or not a Product is produced depending on a Task Unit's Decision. (In the examples outlined in this document, a Product that is not dependent upon the Task Unit's Decision would appear below the Task Label. A Product that was contingent on the Decision would follow the appropriate path from the Decision.)
- Graphical notation should allow the designation that a given product is only an update or revision to a previous Product from a different Task Unit and not a new and separate product. This notation should include the labeling of the location at which the initial or previous version of the Product was created.
- The user shall be able to directly navigate from a Product to all other Task Units in which the Product is updated or revised.
- The user shall be able to directly navigate from a Product to all other Task Units in which the Product is used as an Information Requirement.
- When entering details about a Task Unit, the user should be able to (but not required to) select Products from a list of existing Information Requirements and Products for other Task Units. The existing Information Requirements and Products shall be sortable by name, diagram, and date created. If an existing Product is selected, the new Product shall be identified as an update or revision to the previous Product.

Decisions. Decisions are used to route the operational sequence and task flow within a diagram. An individual Task Unit may include multiple Decisions arranged sequentially and/or in parallel. Unlike Information Requirements and Products, Decisions may also be employed separately from Task Units for the purposes of routing operational or task flow based on external events and conditions or if the decision is internal to the system being described but is of sufficient simplicity to not be described or decomposed in any greater level of detail.

- Decisions shall be graphically associated with the Task Unit of which they are a part.
- A single Task Unit may include multiple Decisions arranged in series and/or parallel. The positioning and layout of the decisions shall clearly show whether the decisions are in series or parallel.
- Additional Decisions may be added to a Task Unit and the arrangement of the Decisions (their order, arrangement in series or parallel) shall be modifiable without additional changes to any portion of the Task Unit.
- The Decision shall graphically show the different routes that may be taken as an outcome of the decision, and each potential outcome shall be labeled.
- Different types of Decisions shall be supported, including basic Yes-No questions, selection from multiple options, and task completion criteria. Each of these types of decisions shall have the same graphical format.

- The user shall be able to select that multiple options from a Decision may be followed simultaneously, and this type of Decision shall be graphically discriminable from those allowing only a single outcome.*

Actions

In some instances, there is little to gain from explicitly identifying the products or information requirements associated with an activity. The activity may be simple and routine enough that there are no information requirements or decisions, or the activity may not be critical to the analysis and therefore not require elaboration. These "non-decomposed tasks" are referred to as Actions. In the current revision of OSD symbology and notation, an Action is identical to a Task Label, with the addition of the word "Action:" at the start of the text field. Note that Task Labels exist only as part of Task Units, and Actions exist only separate from Task Units.

- Actions shall be easily distinguished, though either layout or graphical differences, from Task Labels and Task Units.
- Once an Action has been defined, a user shall be able to expand or decompose the Action to a Task Unit without being required to re-enter any information. The Action itself shall correspond to the Task Label within the new Task Unit.

Triggers

Task Units and events within the documented task flow may at times be initiated due to external events or conditions. The initiating event may be periodic, or it may depend on factors that were not defined due to either scope of the description being generated or a desire to simplify the information documented.

- The labeling of the Trigger shall be a shorthand description of the event or condition that initiates the following Task Unit or other object.
- The User shall be able to record a longer and more detailed description of the initiating event or condition that is associated with the Trigger but is not displayed with the graphic representation of the Trigger. The User may use the description field to identify the conditions that cause the trigger to be initiated.
- Due to their functional similarity, the pictorial representation of Triggers should be conceptually similar but graphically discriminable from Routers into a diagram.

Routers

Routers are used to define flow between diagrams within an entire set of operational sequence diagrams. A Router that denotes flow out of a diagram will have a corresponding Router that denotes the point on another diagram where the flow continues. Routers are not typically used to represent flow from one point to another within the same diagram.

* This Decision type has not been formally implemented as part of the previously described symbology and notation. The means of making this Decision type discriminable has yet to be determined.

- The alphanumeric designator of the related diagram shall be shown in the text of the Router.
- The title (full or abbreviated) of the related diagram shall be shown with the Router's graphical representation.
- Due to their functional similarity, the pictorial representation of Routers into a diagram shall be conceptually similar to that of Routers out of a diagram.
- The designator and title listed with the Router shall be automatically updated if the designator or title is changed in the related diagram.
- When the first Router between two diagrams is created (or deleted), the User shall have the option of automatically creating (or deleting) a second Router in the related diagram.
- As a link is created between two diagrams through the creation of a Router, the User shall be able to designate whether or not that link will always be displayed in the overall wiring diagram of the diagram set. All links shall be displayable on the overall wiring diagram, but the user shall have the capability to toggle the display of those optional links on and off.
- Routers shall support detailed routing to specific Routers within another diagram. The User shall be able to create multiple Routers from one diagram to multiple Routers on a second diagram. Each of the different Routers shall be specifically identifiable with a single Router on the other diagram, and the pairs shall be visually discriminable in the graphic representation of the diagrams. This specification may be part of the Router's text label.
- The User shall have the option of using multiple Routers from a single diagram to go to a single Router on a second diagram. (Users could alternately route the task flow from multiple places in a diagram to the same router, but this capability will permit a less crowded visual representation.)
- When editing or reviewing the diagrams, the Routers shall allow the users to navigate directly to the related diagrams (e.g., through a double-click mouse action or via a pop-up menu option).

Flags

Flags are a representation of "mental notes" or the setting of conditions for use in later decisions. Upon the conclusion of a Task Unit, a Flag may be set to designate the outcome of the Task Unit. Flags can be used similarly to the setting of variables in a computer program.
Note: *Need for this section of requirements is TBD.*

Notes and Annotations

Although the Task Units, their components, and other diagram features will allow the representation of much of the required information to understand the tasks that are documented, amplifying information will also be necessary. This additional information can take the form of both Notes associated with a Task Unit or other element but not displayed on the diagrams or Annotations for specific elements that are displayed on the diagram adjacent

to the element requiring amplification. Notes and Annotations will be useful as placeholders during diagram development, to describe complicated concepts in greater detail, and to record review comments on specific issues or suggestions for the diagrams. Notes will also be useful for transitioning the diagrams to executable models within simulation tools. (*See also the Task Attributes section under Task Support Requirements.*)

- The User shall be able to view both a diagram view, showing all diagram elements and Annotations, and a "Notes" view, showing the diagram view with Notes and other Task Attributes (selectable by the User).
- Both the diagram only and the notes views shall be printable separately.
- The User shall be able to associate Notes with a single element of a Task Unit or a Task Unit as a whole.
- The Diagram View should include a graphical notation that Notes have been entered for a particular part of the diagram.
- For both Notes and Annotations, there shall be both amplifying types and review or comment types, depending on an overall setting for the software tool. When used for diagram creation, all Notes and Annotations shall be of the amplifying type. When used for diagram review, all Notes and Annotations shall be of the review and comment type.

Numbering

During creation and review of a diagram, analysts and reviewers will often refer to entire diagrams or their components by an abbreviation or assigned number. A diagramming tool should support the continued use of such nomenclature. Although components of an electronic version of a diagram can be easily and even automatically renumbered, routine renumbering will prevent users from employing commonly used shorthand designations for diagrams and their components. For final versions and major reviews, however, consistent and intuitive numbering schemes are preferable. The ability to automatically renumber diagrams and their components should be available, but it should also be optional.

- Each Task Unit shall have an automatically generated alphanumeric designator. The designator shall denote the individual diagram within which the Task Unit resides and distinguish it from other Task Units.
- All diagram components other than Task Units (including Actions, stand-alone Decisions, Routers, and Triggers) shall have automatically generated alphanumeric designators that denote the individual diagram within which they reside.
- Each component of the Task Unit shall have its own alphanumeric designator, and different types of components (Information Requirements, Products, Decisions, Task Labels) shall be recognizable by their designators alone.
- Altering the order or sequence of Task Units and other objects shall not automatically change the alphanumeric designators of any objects within the diagram.
- Users shall have the ability to automatically renumber all objects within a diagram or set of diagrams, with the new designators being ordered by order of appearance within the individual diagrams.

- Cutting or copying and pasting an object from one diagram into another diagram will cause its alphanumeric designator to change to that which would have been given to the next newly created object of the same type in that diagram.

Hierarchy of Diagrams

For purposes of both creation and review, operational sequence diagrams must be divisible into sections. Two basic methods of subdividing the overall diagram set are possible. First, individual diagrams may represent different pieces of the overall diagram. In this case, each individual diagram is a separate portion of the overall diagram and the different diagrams represent divisions in flow. An overall wiring diagram is then used to show how the total set of diagrams fits together. Second, the diagrams may be hierarchical decompositions, with one group of "child" diagrams providing more detail about a single "parent" diagram. In this case, a diagram may contain both Task Units and other objects each representing a diagram of multiple Task Units at a more detailed level.

- The User shall be able to divide the diagrams into different sections that may be viewed, edited, saved, and distributed individually.
- The User shall be able to contain the set of diagrams for a single project in a single data file for ease of distribution and management.
- The User shall be able to organize the different sections of the diagrams in either a flat structure with an overall "wiring diagram" showing flow between diagrams or hierarchically with a "child" diagram represented by a single object within a "parent" diagram.*
- An individual diagram set shall be organized via a wiring diagram or hierarchically; methods of organization shall not be mixed within a diagram set.
- The User shall select the organization option (hierarchical or wiring diagram) for a set of diagrams when the set is first created. When the set of diagrams is being edited, reviewed, or expanded by multiple Users, the organization method shall be fixed.
- Different numbering schemes may be applied based on whether hierarchical or wiring diagram organization methods are employed.

TASK SUPPORT REQUIREMENTS

In addition to requirements for what information is recorded in association with OSDs, there are requirements for what capabilities an OSD tool must provide to enable Users to employ it efficiently and effectively.

Diagram Creation

Diagram creation may be performed by the Analyst with extensive experience with the tool and similar diagramming techniques, or it may be performed by an SME with no

* These two different methods may impact the ability to transfer information back and forth with other tools. Further investigation is appropriate.

prior experience. The tool must support the differing backgrounds and skills of the range of potential users.

- Each major graphical element (e.g. Task Labels, Products, and Triggers, but not connecting arrows or lines) shall be of consistent shape and size.
- The User shall be able to drag and drop as well as copy/cut and paste Task Units and individual elements within and between diagrams of a diagram set.
- The User shall be able to automatically center and align diagram elements.
- The User shall be able to specify details of a series of Task Units through a knowledge elicitation capability, such as would be provided by responding to a series of structured questions.
- The User shall have the option of creating blank diagrams directly from an overall wiring diagram.
- The User shall be able to copy entire diagrams as a basis for creating an additional diagram.
- Arrows and lines between elements shall be automatically connect to elements at their endpoints and shall automatically reroute around other elements. The User shall be able to override arrow and line placement by specifying corner or anchor points.

Task Attributes

Depending on the reason for constructing a diagram set, additional information associated with individual Task Units will need to be recorded. This information may be part of the analysis performed by constructing the diagrams themselves, or the information may be required once the information has been transitioned to another design tool. Task Attributes may need to be defined for estimating the workload of tasks, identifying a proposed allocation (to a human or to automation) of a task,* or to document the time expected to be required to accomplish a task. Task Attributes may also be employed to record information that will be available to a Modeler for subsequent use in specifying details required to create an executable model.

- All Task Attributes should be associated with the Task Label of the Task Unit, providing a ready means for the User to access the attributes.
- The predefined Task Attributes available in creating the diagrams should be specified at the initial creation of the diagram set. Eliminating the default display of attributes that are irrelevant for the intended use of the diagrams will permit Analysts and SMEs to focus attention on important attributes.
- The set of predefined Task Attributes should include but not be limited to time to complete a task, statistical distribution (type and value) of completion time, error rates, workload associated with the task, recommendation for allocation of the task, and expected skills or type of user required to complete the task.

* Note that, typically, functions are activities that have not been allocated in any way, and tasks have a specific allocation to a human, automation, or some combination. Under that nomenclature, operational sequence diagrams can be used to document functions, tasks, or functions for assignment as tasks. For the sake of simplicity, the word "task" is always used in this document even though "function or task" may be more appropriate.

- The User shall have the ability to create additional custom task attributes.
- Each Task Unit and Action shall be able to have workload estimates assigned.
- The type of workload measures available shall be flexible, supporting different theories and methods of defining workload.
- The type or types of workload measures available for a given project shall be selected when the project is started, providing a common subset of workload measures for all analysts and reviewers for the project. Other workload measures shall be available but not displayed by default.
- Workload levels assigned to individual Task Units or Actions shall be shared with other design tools.
- The tool shall be able to provide output of Task Attributes in a manner that allows a Modeler to refer to the information once the diagrams are transitioned to another design tool, possibly as a separate data file with references to individual tasks.
- The User shall be able to input both a proposed allocation of a task and a text-format rationale for the allocation.
- When assigned, the proposed automation or manual assignment of a particular task shall be apparent in the graphical view of the Task Label or Task Unit. (Automated tasks and decisions have traditionally been denoted by double-lined symbols.)
- The User shall be able to highlight and copy text from the tool into standard word processing and presentation software packages. Ideally, the User will also be able to directly copy graphical elements, but a screen capture capability may be used instead to replicate graphical elements.

Diagram Review

Diagram review encompasses both the validation of diagrams created initially as operational sequence diagrams and the assessment of processes or information created in a different design tool but translated into OSD format for review by SMEs. Compared to the initial creation of the diagram set, it is likely that more time and effort will be expended on reviewing the diagram set for completeness and accuracy. Many of those who review a diagram or set of diagrams may not have created the diagrams in question or have even interacted with operational sequence diagrams in the past. Review may be performed by a single User or group of Users (working separately on the same initial files), and it may be performed electronically within the tool or based on paper printouts or reports produced by the tool. (*See also Notes and Annotations under the Information Representation Requirements section.*)

- Color shall not be a critical feature in achieving an understanding of an OSD and its components.
- Printouts of diagrams shall be fully understandable when printed or photocopied in black and white.
- Printing and reporting capabilities of the tool shall include output of information in tab-delimited of CSV format, listing names, alphanumeric designators, and other information associated with individual diagram elements. (Note: spreadsheet output will be particularly useful in assigning workload values to tasks.)

- The diagram and notes view may spread out or otherwise distort graphical layout between (but not within) Task Units.
- The tool shall support both "create/edit" and "review" modes of operation. Changing the mode of operation will change default type for new Notes and Annotations and will also allow the User to prohibit changes to layout and content of the diagrams.
- The tool shall provide the User with assistance compiling comments and suggested revisions for a single initial diagram set received from multiple reviewers.
- The User shall be able to serially review each comment or suggested change within a diagram set.
- The User shall be able to automatically generate a report that details all suggested changes or amplifying information added since the last "baseline" of the diagram set.

Import and Export of Information

One of the primary benefits of tool support for the creation of operational sequence diagrams is an opportunity to share the information with other design tools. Information that is entered by SMEs could be used as the starting point for simulations in more complex and capable design tools, obviating the need to re-enter data by hand. Additionally, an import capability would allow information from complex design tools to be readily reviewed by SMEs, providing better feedback faster. Optimal integration of information will only be accomplished if the data requirements and data fields of the external tools are taken into consideration.

- The capability to import and export information to share with other tools may be an integral part of the tool or those functions may be allocated to some intermediary tool (such as Interchange SE).
- The tool shall be able to import information from modeling tools, converting the information to a reviewable format.
- The tool shall be able to import a list of Task Units (minimally Task Labels) or Actions from either tab-delimited or CSV formats.
- Information imported from other tools shall be modifiable by the User, allowing the addition of information and attributes not part of the original imported information set.
- The tool shall be able to export information to modeling tools, providing relevant information necessary for further detailed modeling.
- The tool shall be able to export information in XML format.

Training

Training requirements were one of the key motivators in the final revision of the OSD symbology. Although the initial organization of the Task Units and other elements was easier to read and comprehend than other diagramming techniques that were available, the training burden was still too great at times. In a situation where multiple SMEs are brought together for a short review period, the time spent in training and orientation becomes time unavailable for review of the diagrams and associated information.

The user interface of the tool should be designed such that training for new users of the tool does not require an excessive amount of time. Expected characteristics of the user population are outlined in the Intended Users section. The development of the tool shall include periodic usability testing to ensure that the following goals are expected to be reached:

- Users should be able to effectively employ the tool to electronically review existing diagrams after less than twenty minutes of orientation and training.
- Users should be able to effectively employ the tool to create diagrams after less than one hour of orientation and training.
- Users should be able to effectively employ the tool to identify and document Task Attributes after less than two hours of orientation and training.
- The tool should provide Users with an optional, brief, interactive tour and tutorial on its features and usage.
- Basic help functionality of the tool shall include information on the meaning of all symbols and elements of the diagrams. This information shall be easily accessible and easily printed out.
- The tool's help functionality shall specify the data field(s) in other design tools to which the diagram elements and other information in the tool correspond. The list of relevant tools shall include those tools with which the OSD tool is able to share data via import or export.

REFERENCES

1. Kurke, M.I., "Operational Sequence Diagrams in System Design," *Human Factors*, Vol. 3, 1961, pp. 66-73.

2. *Human Engineering Program Process and Procedures*, MIL-HDBK-46855A, 17 May 1999.

3. Chapanis, A., *Human Factors in Systems Engineering*, Wiley-Interscience, New York, NY, 1996.

4. Woodson, W.E.; Tillman, B.; and Tillman, P, *Human Factors Design Handbook*, Second Edition, McGraw-Hill, New York, NY, 1991.

5. Van Cott, H.P. and Kinkade, R.G., Eds., *Human Engineering Guide to Equipment Design*, Wiley-Interscience, New York, NY, 1972.

6. Wallace, D.F.; Bohan, J.A.; and Perry, A.A., "Systems Engineering Processes Applied to Human Engineering: Requirements Definition and Design Traceability Using the TIGERS Method," Workshop presented at the HFES 1997 Annual Meeting, Albuquerque, NM.

7. Wallace, D.F.; Winters, J.J.; and Lackie, J.H., "An Improved Operational Sequence Diagram Methodology for Use in System Development," in *Proceedings of the Human Factors and Ergonomics Society*, San Diego, CA, 2000, pp. 6-505 – 6-508.

DISTRIBUTION

	Copies
DOD ACTIVITIES (CONUS)	
ATTN CODE A76 (TECHNICAL LIBRARY) COMMANDING OFFICER CSSDD NSWC 6703 W HIGHWAY 98 PANAMA CITY FL 32407-7001	1
DEFENSE TECHNICAL INFORMATION CENTER 8725 JOHN J KINGMAN ROAD SUITE 0944 FT BELVOIR VA 22060-6218	2
ATTN PEO(S)-M (BOB BOST) 1333 ISAAC HULL AVENUE SE STOP 2201 BLDG 197 ROOM 3W-1241 WASHINGTON NAVY YARD WASHINGTON DC 20376-2201	2
ATTN GWENDOLYN CAMPBELL COMMANDING OFFICER NAVAIR ORL TSD CODE 4961 12350 RESEARCH PARKWAY ORLANDO FL 32826-3224	2
ATTN GLENN OSGA COMMANDING OFFICER SPACE AND NAVAL WARFARE SYSTEMS CENTER SAN DIEGO CODE D441 52345 PATTERSON ROAD SAN DIEGO CA 92152-7150	1
ATTN NANCY DOLAN OFFICE OF THE CHIEF OF NAVAL OPERATIONS 2 NAVY ANNEX CODE N125G WASHINGTON DC 20370	1
ATTN LAUREL ALLENDER US ARMY RESEARCH LAB ATTN AMSRL-HR-MB ABERDEEN PROVING GD MD 21005-5425	1
ATTN JAN DICKIESON OFFICE OF NAVAL RESEARCH 800 N QUINCY STREET ARLINGTON VA 22217-5660	1
ATTN SAM NICHOLSON OFFICE OF NAVAL RESEARCH 800 N QUINCY STREET ARLINGTON VA 22217-5660	1
NON-DOD ACTIVITIES (CONUS)	
THE CNA CORPORATION 4825 MARK CENTER DRIVE ALEXANDRIA VA 22311-1850	1
ATTN JOHN WINTERS BASIC COMMERCE & INDUSTRIES P O BOX 1748 DAHLGREN VA 22448	4
ATTN JOHN LACKIE SONALYSTS INC P O BOX 1839 DAHLGREN VA 22448	2
ATTN CATHY PERRY ACS 16539 COMMERCE DRIVE SUITE 10 KING GEORGE VA 22485-5806	2
ATTN DAVE DAHN MICRO ANALYSIS & DESIGN INC 4900 PEARL EAST CIRCLE #201E BOULDER CO 80301	1

DISTRIBUTION (Continued)

		Copies
INTERNAL		
B60	(TECHNICAL LIBRARY)	3
G50	(DANIEL WALLACE)	3
N05M	(TRISH HAMBURGER)	3
N93	(DENNIS WHITE)	1
B05	(HARRY CRISP)	1
B35	(NGOCDUNG HOANG)	1